

Advanced Regression 4b: Machine learning, ensemble methods

Garyfallos Konstantinoudis

Mar 14, 2025

Ensemble methods:

- Bagging
- Random forests
- Boosting
- Variable importance
- Ensemble methods in R

Bootstrap aggregation (bagging)

- Let the original dataset contain n samples
- Take $n' < n$ repeated samples (with replacement) of the original dataset to create B new datasets ($b \in 1, \dots, B$ bootstrapped training datasets).
- Grow a tree on each bootstrapped training datasets.
- Each of the trees has little bias, but high variance.

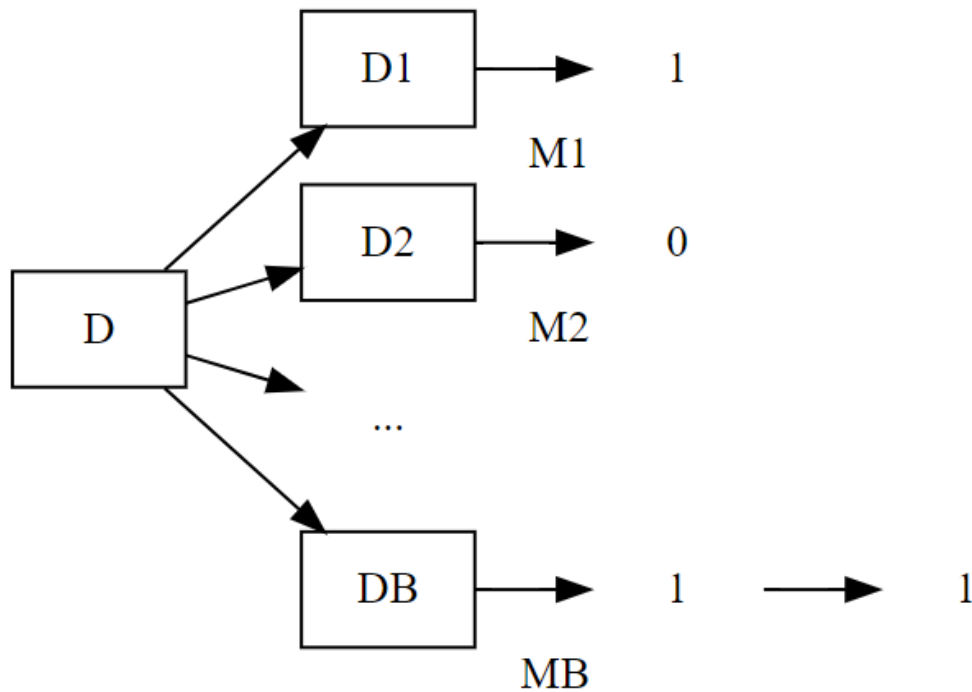
Average over trees

- Regression tree: Compute the mean over the B predictions f_b of each individual regression tree

$$f_{bag}(x) = \frac{1}{B} \sum_{b=1}^B f_b(x)$$

- Classification tree: Majority vote (mode), the final prediction is the one that occurred most frequently among the B trees

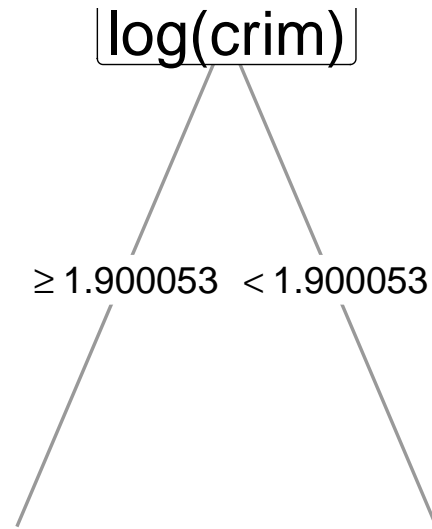
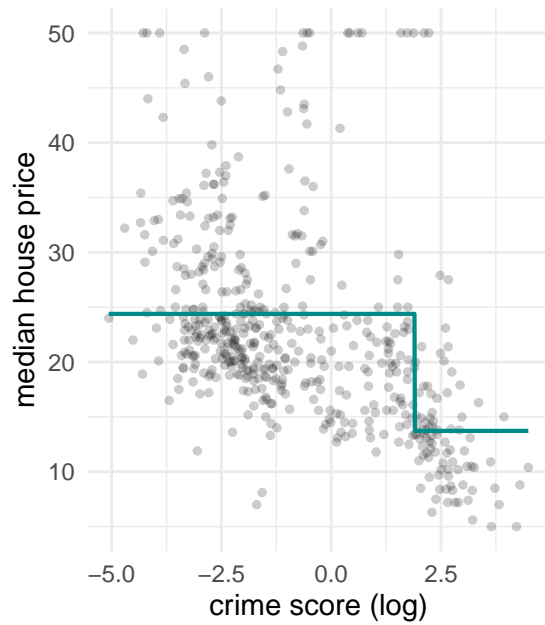
Bagging



Example (taken from Prof Filippi lecture)

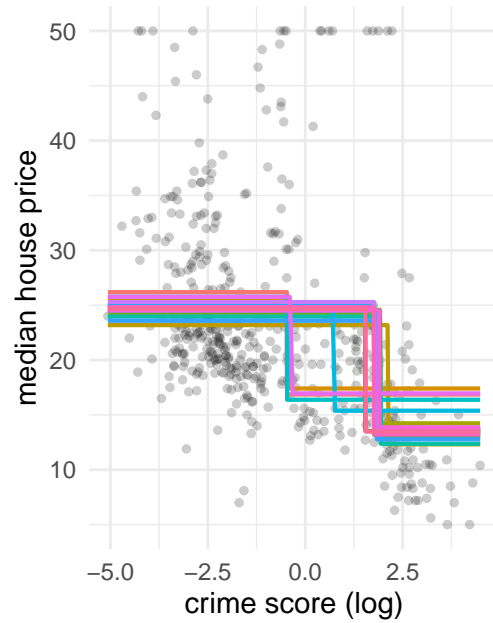
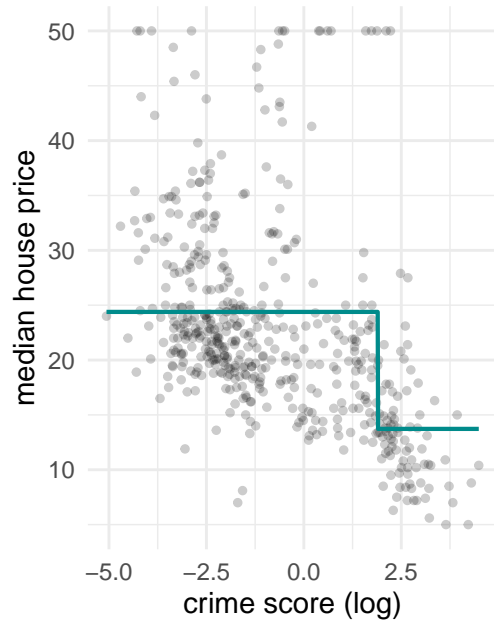
- Aim: predict median house prices (Boston) based only on crime rate
- Consider a tree with a single split at the root

Warning: Using ``size`` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use ``linewidth`` instead.



Example (taken from Prof Filippi lecture)

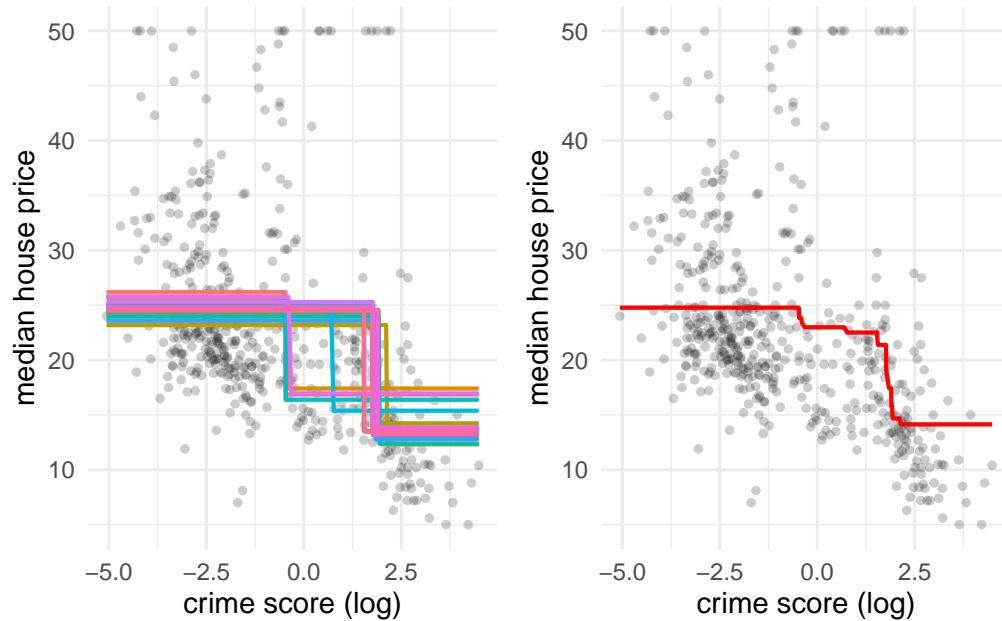
20 bootstrap samples



Example (taken from Prof Filippi lecture)

Summarise the samples

Joining with ``by = join_by(row)``



Out-of-bag error

- Bagging has an intrinsic way of evaluating prediction performance

Out-of-bag samples

- The bootstrapped training dataset contains n samples drawn at random
- $n - n'$ samples are left out. They are the out-of-bag samples.
- The out-of-bag error can be used to estimate the test error and prediction performance.
- No cross-validation is needed.

Random forest

- Bagging always considers all p variables to build a decision tree.
- Consequently, individual trees in bagging may look very similar.
- Random forest selects at random a subset of m variables to be considered at each split.
- Consequently, individual trees in random forests may look very different (or random).

How to select m ?

- Regression random forest: $m = p/3$
- Classification random forest: $m = \sqrt{p}$

- Bagging: $m = p$



(a) Bagging



(a) Random forest

Boosting of trees

- Trees are grown sequentially, using information from previously grown trees.
- Boosting learns from mistakes, after fitting the first tree, further trees are grown based on the residuals.
- Tuning parameters:
 1. B : Number of trees
 2. λ : Learning rate
 3. d : Number of splits per tree, interaction depth

Boosting algorithm

1. Initialise the output parameters: the predicted values $f(x) = 0$ and the residuals $r_i = y_i$ for all observations i .
2. Loop through $b \in 1, \dots, B$, repeat:
 1. Fit tree b with d splits, obtain new predicted values $f_b(x)$
 2. Update

$$f(x) = f(x) + \lambda f_b(x)$$

3.

$$r_i = r_i - \lambda f_b(x)$$

3. Output as the final model the boosted predictions

$$f_{boost}(x) = \sum_{b=1}^B \lambda f_b(x)$$

Outlook: gradient and XGboost

- Gradient Boosting: A special case of boosting where errors are minimized by gradient descent algorithm.
- eXtreme Gradient Boosting (XGBoost):
 - Boosting algorithm, sequential learning
 - Approximation and regularization
 - Computationally efficient, based on parallel and distributed computing

Variable importance: Understanding how algorithms work

Interpretable machine learning

- Variable importance in ensemble trees allows to rank variables by their importance in the model.

Variable importance: Understanding how algorithms work

There are two approaches for variable importance:

1. **Permutation:** Mean decrease in accuracy

- First fit the model on the original data and evaluate the prediction error on the out-of-bag samples.
- Permute variable j , refit the model and re-evaluate the prediction error on the out-of-bag samples.
- The mean decrease of accuracy after permuting variable j can be used to measure the importance of the j th variable.

2. **Gini-index:** Mean decrease in impurity

- Record the decrease of impurity every time a tree is split at variable j .
- Average over all trees.

Bagging and random forests in randomForest

```
randomForest(  
  x, y, # train data  
  xtest = NULL, ytest = NULL, # test data  
  ntree = 500, # number of trees to grow  
  # number of variables to consider in each tree  
  mtry = if (!is.null(y) && !is.factor(y)) max(floor(ncol(x) / 3), 1) else floor(sqrt(ncol(x)  
  replace = TRUE, # sample with replacement  
  importance = TRUE # computes variable importance  
)
```

Bagging and random forests in randomForest

mtry is number of variables to consider in each tree

- Regression random forest: $m = p/3$
- Classification random forest: $m = \sqrt{p}$
- Bagging: $m = p$

```
data(BostonHousing)  
  
# here, use formula rather than x and y  
fit <- randomForest(medv ~ ., data = BostonHousing)  
fit
```

Call:

```
randomForest(formula = medv ~ ., data = BostonHousing)  
      Type of random forest: regression  
      Number of trees: 500
```

No. of variables tried at each split: 4

```
      Mean of squared residuals: 10.27921  
      % Var explained: 87.82
```

Bagging and random forests in randomForest

- \$predicted: Predicted values
- Classification measures: \$err.rate, \$confusion, \$votes

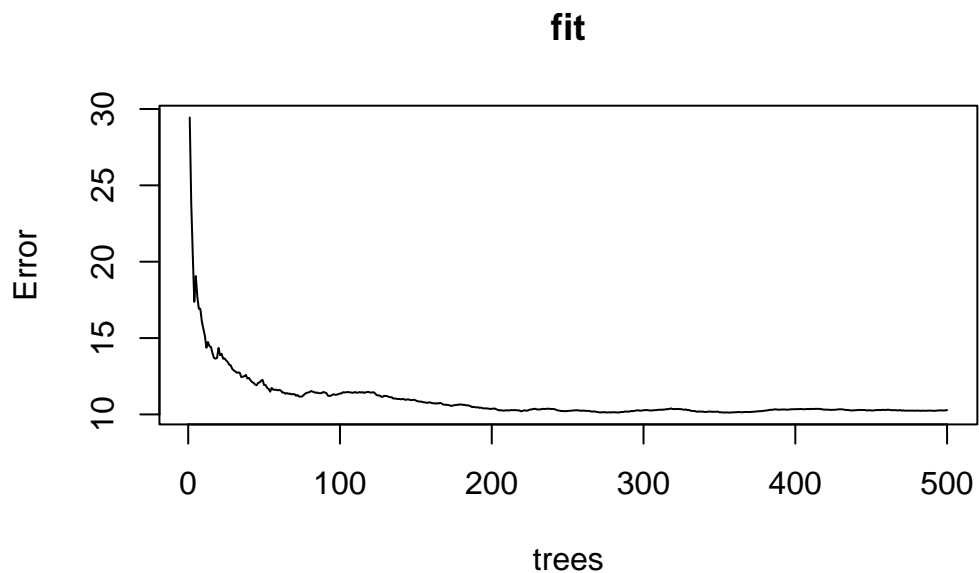
- Regression measures: `$mse`, `$rsq`
- `$importance`: Variable importance (mean decrease in accuracy and mean decrease in impurity)

Further functions

- `predict.randomForest`: Predict new data
- `plot.randomForest`: Plot error against number of trees
- `varImpPlot`: Plot variable importance

Bagging and random forests in `randomForest`

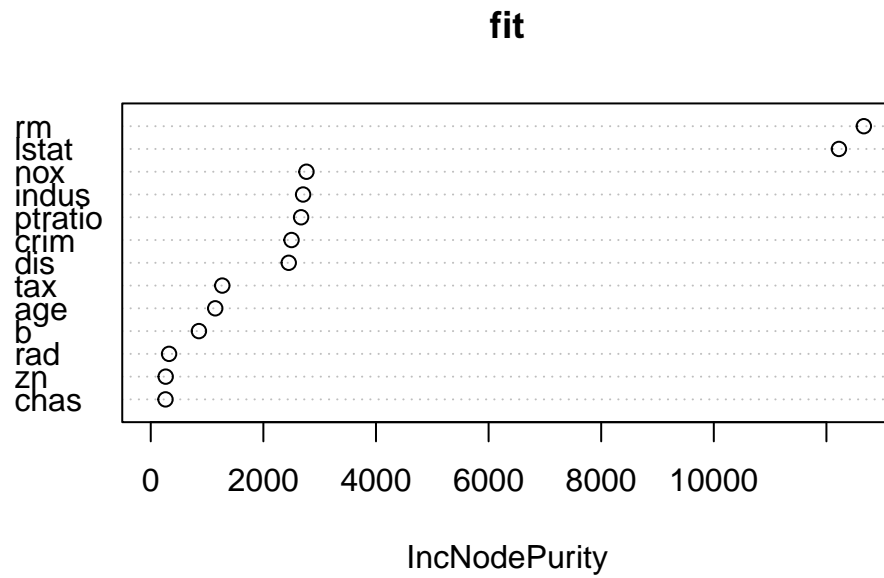
```
plot(fit)
```



Bagging and random forests in `randomForest`

Mean decrease in impurity

```
varImpPlot(fit)
```



Boosting in gbm

```
gbm(
  formula,
  distribution, # family of the outcome
  data,
  n.trees = 100, # number of trees to fit
  interaction.depth = 1, # maximum depth of each tree
  n.minobsinnode = 10, # minimum number of observations in the terminal nodes
  shrinkage = 0.1, # shrinkage factor
  cv.folds = 0 # performs additional cross-validation
)
```

Further functions

- `predict.gbm`: Predict new data
- `summary`: Relative influence

Take away: Ensemble methods

- Decision trees offer great interpretability.

- But they tend to overfit, performing poorly in prediction.
- Ensemble methods like bagging, random forest and boosting, fit many trees in different variations and average over them.
- This reduces the variance and leads to greater generalizability.
- Variable importance measures help to understand the contribution of specific variables